International Journal of Geo-Information

MDPI

Article

Automatic Geodata Processing Methods for Real-World City Visualizations in Cities: Skylines

Jan Pinos *, Vit Vozenilek and Ondrej Pavlis

Department of Geoinformatics, Faculty of Science, Palacký University Olomouc, 17. listopadu 50, 77146 Olomouc, Czech Republic; vit.vozenilek@upol.cz (V.V.); on.pavlis@seznam.cz (O.P.)

* Correspondence: jan.pinos01@upol.cz; Tel.: +420-608-158-858

Received: 5 November 2019; Accepted: 30 December 2019; Published: 1 January 2020

Abstract: The city-building game Cities: Skylines simulates urban-related processes in a visually appealing 3D environment and thus offers interesting possibilities for visualizations of real-world places. Such visualizations could be used for presentation, participation, or education projects. However, the creation process of the game model from geographical data is inaccurate, complicated, and time consuming, thus preventing the wider use of this game for non-entertainment purposes. This paper presents the automatic methods scripted in the Cities: Skylines application programming interface (API) and bundled into a game modification (commonly referred to as a game mod) named GeoSkylines, to create a geographically accurate visualization of real-world places in Cities: Skylines. Based on various geographical data, the presented methods create road and rail networks, tree coverage, water basins, planning zones, buildings, and services. Using these methods, playable models of the cities of Svit (Slovakia) and Olomouc (Czech Republic) were created in the game. The game mod GeoSkylines also provides methods for exporting game objects such as roads, buildings, and zones into a Geographic Information System (GIS) data format that can be processed further. This feature enables the game Cities: Skylines to be utilized as a data collection tool that could be used in redevelopment design projects.

Keywords: geodata; visualization; game mods; city-building games; Cities: Skylines

1. Introduction

Games, and video games in particular, are immensely popular, as documented in a recent survey by the Pew Research Center. According to this study, 90 percent of surveyed teenagers play video games of some kind either on a cell phone, computer, or a console [1]. Though games are mainly for entertainment, they have also been a subject of research in numerous studies across various academic fields, especially in education due to their ability to pass on information with ease [2,3]. In the field of urban planning, video games are often used for civic engagement [4–6]. City-building games represent a specific example of game use in the field of urban planning. Aside from inspiring new generations of city planners [7,8], these games have been used primarily in education, but have also recently begun to be utilized for visualizations of real-world places in participation projects [9–12]. However, the lack of professional tools and methods for processing geographical data often prevents the efficient use of video games for non-entertainment purposes. This problem also applies to Cities: Skylines. The possibilities of uploading geographical data into the game, or extracting the game objects into geographical data, are quite limited. Authors of previous case studies, where Cities: Skylines was used for visualizations of a real-world place, have reported that the creation process of the game model from geographical data is complicated and lengthy. Nevertheless, Cities: Skylines offers an extensive application programing interface (API) that allows for the game to be modified from its looks to its behavior. Using the API, a game modification (often referred to as a game mod) for processing

geographical data can be scripted. Such a tool can then help remove the stumbling block that has prevented the game Cities: Skylines from being more widely adopted for non-entertainment purposes.

1.1. Potential and Limitations of City-Building Games

In general, as argued above, commercial video games are highly seductive to almost anyone. Nevertheless, using them "seriously" (for purposes that are not primarily for entertainment) has its constraints. Rufat and Minassian [13] compared selected city-building games (SimCity 4 and CityLife) with urban modeling tools used in research and concluded that simulations in the city-building games were based on similar models to the ones used in research, but the main difference was in the ability to change the simulation parameters after observation and learning. Whereas scholarly modeling tools offer this feature, games do not. This is largely due to the protection of intellectual property. In other words, simulation logic in games is a "black box", where the rules can be deduced only through exploration. This problem was also highlighted by Devisch [14], who studied the possible use of SimCity in planning processes. SimCity has also been studied from the point of view of urban hydrology, but understanding the underlying algorithms is challenging, since it can only be done by running trial simulations due to the "black box behavior" of the game [15].

City-building games have also been criticized for their simplification of urban processes. For example, the lack of a mixed zone usage, a concept seen as beneficial among urbanists [16], was spotted in early versions of SimCity [17] and later in SimCity 4 and Cities: Skylines [18]. City-building games are played from the position of an all-powerful mayor, though in reality, the planning process involves many stakeholders [18,19]. The critique of these simplifications, however, can be seen as unfair. Urban planning processes are very complex and as such, cannot be conclusively modeled, even with scholarly tools [20]. Many simplifications have been caused by our inability to comprehend and describe the full complexity of urban systems. Nevertheless, some simplifications have been incorporated into games simply to enhance the playability of the game, thus making it more appealing to the consumer [21].

Many of the obstacles and restrictions of a commercial game such as the ones presented above can be overcome or at least mitigated by incorporating changes to the game's behavior. The process of changing the video game using custom scripting is known in the gaming community as "modding" [22]. Modding is available in most modern video games as it is a feature desired by the gaming community [23]. However, the level of extensiveness and support varies from game to game. Whereas in some cases the modding possibilities are quite limited, in other cases, the game's behavior or looks can be changed significantly. The latter is true for Cities: Skylines, where players can write custom scripts known as "mods" to adjust the game, its looks, or behavior. Mods can range from cosmetic (e.g., adding a new type of tree to the game) to complex changes of the game's behavior (e.g., traffic management). The possibility of modifying the game is highly appreciated by the gaming community: 175,970 Cities: Skylines mods were created by the beginning of 2019 [24].

The utilization of the agent-based model simulating citizens and vehicles in city-building games might also offer interesting possibilities. Interaction between agents creates an artificial society where real-life phenomena emerge [25]. Employing the agent-based model, the player of Cities: Skylines can build a city that is close to the chaotic, unpredictable, and self-organizing system that defines modern cities [26]. A model of a real-world city created in Cities: Skylines could then be used for learning about the urban processes taking place in the modeled city: zone demand, land value evaluation, levels of noise or waste pollution, access to services, overall happiness of the citizens, traffic management, and many others. Additionally, by using the game's features, new parts of the modeled city (e.g., a suburb) can be built or an existing part rebuilt (e.g., changing a problematic intersection into a roundabout) and the effects of such changes can quickly be explored and evaluated.

1.2. City-Building Game Genre Overview

The best-known city-building game is undoubtedly SimCity, from the development studio Maxis. Moss [7] walks us through the history of the city-building game genre from the original SimCity released in 1989, over various titles in this genre to the latest release in the SimCity series: SimCity 2013. Aside from being commercially successful, SimCity has also been recognized in the academic field, most notably as an experimental learning tool in spatial planning classes [27–30]. However, SimCity 2013—the most recent title in the SimCity series—encountered several problems after its release, which led to the game's poor reception among players [7]. In 2015, the Finnish game development studio Colossal Order released their take on a city-building game, named Cities: Skylines. This game was well received as it delivered desired features such as a 3D graphics environment, an extensive API for creating game modifications, a mass transit system, and well-functioning agent-based simulations controlling citizens, vehicles, and other game objects [7]. Today, Cities: Skylines is arguably the best city-building game on the market. We can conclude this from (1) its commercial success, where Cities: Skylines has sold six million copies [24] compared to SimCity 2013's two million copies [31]; and (2) the size of the gaming community gathered on forums on Reddit.com where the forum dedicated to Cities: Skyline r/CitiesSkylines has more than 200,000 subscribers [32] whereas the forum dedicated to SimCity's latest reincarnation, r/SimCity is followed by a much lower number of 25,000 subscribers [33].

1.3. Cities: Skylines Overview

Cities: Skylines can be described as a single-player open-ended city-building simulation. In the game, players engage in urban planning by establishing the road network, controlling zoning, providing public services and public transportation, and taxation. Players maintain various elements of the city such as its budget, education, employment, pollution levels, etc. All of the simulated phenomena can be monitored in 29 "info views" that provide visually attractive outputs [34]. Figure 1 presents an example of an info view displaying the noise pollution levels in the modeled city.

Figure 1. Example of a Cities: Skylines info view displaying the simulated noise pollution [34].

As above-mentioned, Cities: Skylines can be adjusted significantly thanks to the rich modding environment. This game offers an extensive API written in the programming language C# as well as online support in the form of modding documentation and user forums. Numerous mods created by the gaming community have been introduced to the game that modify simulations related to urban planning: mods that incorporate more realistic traffic patterns (e.g., traffic jams in the morning and afternoons when people go to and from work/school); mods that allow for the preservation of historical buildings, the lack of which was criticized by Bereitschaft [18]; and even mods that allow vertically combining commercial and residential zones. Table 1 lists selected Cities: Skylines game mods that work with geographical data or modify the simulated urban processes.

Table 1. Selected Cities: Skylines "urban" mods.

Mod Name	Mod Description
Traffic Manager: President edition	Modifies the traffic simulation logic. Enhances vehicle agents AI (e.g., enforcing parking). Adds new possibilities of interaction such as the timing of traffic lights, changing the right of way or reserving lane for specific type of vehicle.
Real Time	Enables the realistic changing of day and night and modifies the behavior of the agents accordingly.
Realistic population and consumption	Modifies attributes of game objects, e.g., the number of residents living in buildings, which are exaggerated in the original version.
Any Road Outside Connections	Allows setting required road utilization by creating roads that are outside of the game area.
Ploppable RICO	Allows user configuration of game object's attributes. This can be used to achieve mixed use buildings or change the effect of the building on its surroundings.
No abandonment; No fires; and similar	These mods (in some cases very simple ones) mitigate unrealistic behavior of the original version (e.g., frequent fires due to exaggerated demand for fire services).
Image Overlay	Allows the uploading of any PNG image as a transparent layer. This can be used to visualize any geographical data.
Cimtographer	Creates road network based on road geographical data from Open Street Map (OSM).
Terrain.party	External tool for obtaining heightmaps that can be uploaded in the game's map editor. Heightmaps are generated from globally available Digital Elevation Models (DEM): SRTM3 v4.1, SRTM30 Plus and ASTER 30 m.

Due to the capability of Cities: Skylines to simulate thousands of citizens and cars, Eisele et al. [35] used this game to model decentralized smart systems. To apply the desired behavior of traffic lights for the simulated scenarios, a mod was created using the game's API [35]. These examples demonstrate the strength of the Cities: Skylines' modding, which is arguably unmatched in the city-building genre.

1.4. Cities: Skylines in Visualization Case Studies

Cities: Skylines is built on the gaming engine Unity, thanks to which it offers a visually appealing graphics environment in a spacious 3D world that can be roamed almost freely. Though the game is primarily oriented toward building imaginary cities, many players have been inclined to use this game to create a model of their hometowns or other well-known real-world places [36]. However, visualizations in Cities: Skylines can also be interesting for the general public.

The city of Hämeenlinna, Finland held a contest to design an area near the city center using the game Cities: Skylines. First, the city planners created a map for the game containing the road network and water resources, which served as a template for the contestants, who then turned this

map into a playable model of the given city area [9]. The city of Stockholm in Sweden used Cities: Skylines to model the intended development of the Royal Seaport district and offered this model to the general public, who could then explore the modeled area and contribute new ideas to the development plan [10]. Another visualization using Cities: Skylines was conducted at the Norwegian University of Life Sciences in Oslo. First, a model of Oslo was created in the game by an experienced player and then this model was examined by students from the point of view of urban planning and modeling [11].

A model of the German city of Braunschweig was created in Cities: Skylines to simulate the production of urban factories [12]. In this research, Juraschek et al. [12] chose Cities: Skylines for its simulating capabilities, visually rich outputs of simulated phenomena such as noise pollution, and the extended possibilities of amending the game with the use of "mods". However, regarding building the model, Juraschek et al. [12] states that: "As for now no automated script for transferring the topological data into the game is available that produces high quality results. This can make the creation of the model very time consuming in the beginning." Therefore, in order to remove the obstacles in the model creation process, this research presents methods that allow for automated, simple, repeatable, and geographically accurate modeling of most of the inhabited locations on Earth in Cities: Skylines.

2. Materials and Methods

An important task for using Cities: Skylines "seriously" is to find a way to efficiently process geodata for the purposes of the game. The current tools have many limitations and using them for model creation is time consuming and complicated. Therefore, the main focus of this research was to provide a tool that will help bridge the gap between the game and the GIS domain. This tool should offer automated, simple to use, repeatable, and geographically accurate methods of importing geodata into the game as well as exporting the game objects out as GIS data.

The presented game mod GeoSkylines was designed to include the following aspects:

- Enable the use of any dataset of geodata (not tied directly to OSM).
- Enable formatting of the source geodata.
- Provide a minimalistic graphics user interface (GUI) so the mod is less prone to breaking after frequent game updates.
- Give maximum focus to the accuracy of the import and export methods.

In order to fulfil the first two aspects, the process of preparing the geodata was done separately, before running the import methods of the GeoSkylines game mod. For storing the geodata, we chose to use a simple Comma Separated Value (CSV) format with geometry data recorded in Well-Known Text (WKT) format. Regarding the third aspect, the GeoSkylines mod does not include a GUI; instead, methods are activated via specified hot key combinations.

The overall process of creating a playable model in Cities: Skylines consists of three stages, as shown in Figure 2.

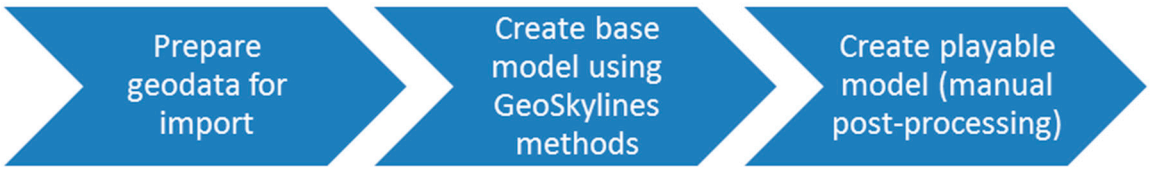

Figure 2. Stages of creating a model of a real-world place in Cities: Skylines using GeoSkylines methods.

First, using standard GIS tools, geodata are downloaded, transformed, and converted into a suitable CSV file. Prepared CSV files are stored in the game's default directory. This directory is also the storage location for CSV files that are created as an export of the game objects, the raster image of tree coverage that is used by the tree import method, CSV files used for matching the types of geographical objects to the types of game objects (e.g., road types), and a configuration file that is used

to define the latitude and longitude coordinates of the selected mid-point and other settings that help adjust the import and export processes.

In the second stage, a base model is created within the game's map editor by running import methods of the GeoSkylines game mod. All methods are described in Table 2. A full description of the presented methods is available on development platform GitHub [37].

Table 2. List of methods of the GeoSkylines game mod.

Method Name	Method Description	Hot Key	Requires
ImportRoads()	Uploads roads_rwo.csv, matches road types according to rwo_cs_road_match.csv and creates a road network.	Ctrl + R	roads_rwo.csv, rwo_cs_road_match.csv, import_export.conf
ImportRails()	Uploads rails_rwo.csv, matches rail types according to rwo_cs_rail_match.csv and creates a rail network.	Ctrl + L	rails_rwo.csv, rwo_cs_rail_match.csv, import_export.conf
ImportWaterBody()	Uploads water_rwo.csv and lowers the terrain for each polygon using the Ray casting algorithm.	Ctrl + W	water_rwo.csv, import_export.conf
ImportWaterWay()	Uploads waterway_rwo.csv and lowers terrain for each waterway segment.	Ctrl + Q	waterway_rwo.csv, import_export.conf
ImportTreesRaster()	Uploads trees.png and for every non-white pixel creates a tree.	Ctrl + T	trees.png (1081 × 1081 resolution), import_export.conf
ImportTreesVector()	Uploads trees_rwo.csv and creates a tree for each record.	Ctrl + V	trees_rwo.csv, import_export.conf
ImportZones()	Uploads zones_rwo.csv, matches locations of each zone with the locations of game's zone blocks and sets the zone.	Ctrl + Z	zones_rwo.csv, rwo_cs_zone_match.csv, import_export.conf
ImportServices()	Uploads amenity_rwo.csv, matches service types according to rwo_cs_service_match.csv and creates services.	Ctrl + S	amenity_rwo.csv, rwo_cs_service_match.csv, import_export.conf
ExportSegments()	Exports all road segments created in the game into a CSV file.	Ctrl + G	import_export.conf
ExportBuildings()	Export all buildings created in the game into a CSV file.	Ctrl + H	import_export.conf
ExportZones()	Exports all zones created in the game into a CSV file.	Ctrl + J	import_export.conf
ExportTrees()	Exports all trees created in the game into a CSV file.	Ctrl + K	import_export.conf
DisplayLLOnMouseClick()	Displays screen, game and WGS 84 coordinates of the place of the click.	Ctrl + left mouse click	import_export.conf
OutputPrefabInfo()	Outputs all road, building and tree types that are currently loaded in the game.	Ctrl + P	nothing

To achieve the best result for the created base model, the methods should be called in this order:

1. Import terrain using the inbuilt heightmaps loader;
2. Create game road network based on geographical road segment data;
3. Optional step: create game rail network based on geographical rail segment data;
4. Lower terrain to form water basins based on geographical water resources data;
5. Create game tree coverage based on geographical tree coverage data (rasterized); and
6. Create game services based on geographical data of services (e.g., OSM amenity data).

The last stage of creating a playable model in Cities: Skylines involves manual post-processing of the base model and turning it into a playable model that can run game simulations. The manual post-processing includes fixing issues of the generated base model (e.g., caused by inaccurate geodata), but also adding necessities such as connecting the city model to a highway (in order for new inhabitants to move in) or adding water resources (creating water basins is automated but placing a water resource must be done manually). The amount of time spent on manual post-processing of the base model

depends on the required level of detail. Doing the bare minimum so the model is playable can take tens of minutes, but in cases where incorporating the greatest details (e.g., adding unique buildings) to maximize the attractivity of the model is desired, the manual post-processing can take hours to complete. Regardless of the case, the manual post-processing and fine tuning of the model should be conducted by an experienced Cities: Skylines player.

2.1. Converting Geographical Coordinates into Game Coordinates

The cornerstone of all import and export methods is the conversion of geographical coordinates into game coordinates (and vice versa). Cities: Skylines uses cells with dimensions of 8 × 8 m; 240 × 240 of these cells make one tile (1920 × 1920 m); 9 × 9 tiles make the maximum game area. This in metric is 17.28 × 17.28 km, for a total area of 298.5984 squared km. However, the playable area in the base version consists of just 5 × 5 tiles (9.6 × 9.6 km) and the maximum area can only be unlocked with the installation of the game mod named "81 tiles".

The game utilizes a projected coordinate system using meters as units. This coordinate system has three axes: x, y, and z, where the y axis, contrary to the geographical standards, stores height values (behavior inherited from the gaming engine Unity). Axis z then serves as the 'northing', according to the GIS conventions. The point of origin is in the center of the game area, thus the axes x and z range from −8640 to 8640.

We can consider the game's coordinate system as a variation of the Universal Transverse Mercator (UTM) projected coordinate system. For the actual coordinate conversion, we can then simply use existing methods for converting WGS 84 coordinates to UTM coordinates. The overall conversion process for import will then follow these steps:

- Choose a 17.28 km × 17.28 km area of the modeled location;
- Calculate the mid-point of this area;
- Convert WGS 84 coordinates of the mid-point to UTM coordinates;
- Convert WGS 84 coordinates of all geographical objects (roads, buildings, etc.) to UTM coordinates;
- Game coordinates of any object created in the game then equal to UTM coordinates of this object minus the UTM coordinates of the mid-point;

$$\boldsymbol{X} = \boldsymbol{E_O} - \boldsymbol{E_M} \tag{1}$$

$$\boldsymbol{Z} = \boldsymbol{N_O} - \boldsymbol{N_M} \tag{2}$$

where $\boldsymbol{X}$ and $\boldsymbol{Z}$ represent the game axes in 2D space; $\boldsymbol{E}$ represents easting; $\boldsymbol{N}$ represents northing; $\boldsymbol{O}$ represents the geographical object; and $\boldsymbol{M}$ represents the mid-point.

Likewise, the conversion process for export will follow these steps:

- Convert WGS 84 coordinates of the mid-point to UTM coordinates;
- Calculate the UTM coordinates of the game objects:

$$\boldsymbol{E_O} = \boldsymbol{X} + \boldsymbol{E_M} \tag{3}$$

$$\boldsymbol{N_O} = \boldsymbol{Z} + \boldsymbol{N_M} \tag{4}$$

- Convert the UTM coordinates of game objects to WGS 84 coordinates (the required UTM zone is obtained from the UTM coordinates of the mid-point).

2.2. Exporting Game Objects as Geographic Information System (GIS) Data

The GeoSkylines game mod also offers methods to export game objects, specifically trees, roads, railways, buildings, and zones. This feature could be used for data collection in participation projects like the ones above-mentioned where players were asked to design a new suburb. Using the export

methods in GeoSkylines, the best designs could be exported into GIS data that could then be further processed. As an example, Figure 3 shows the exported roads and buildings in QGIS. Purple lines represent game roads and grey polygons represent game buildings. Exported data were displayed on top of a base map layer in the location of the city of Svit (Slovakia), which confirmed the accuracy of the used algorithms for converting coordinates. Additionally, as visible in Figure 3, the Bezier curve algorithm was employed to efficiently export curved roads created within the game.

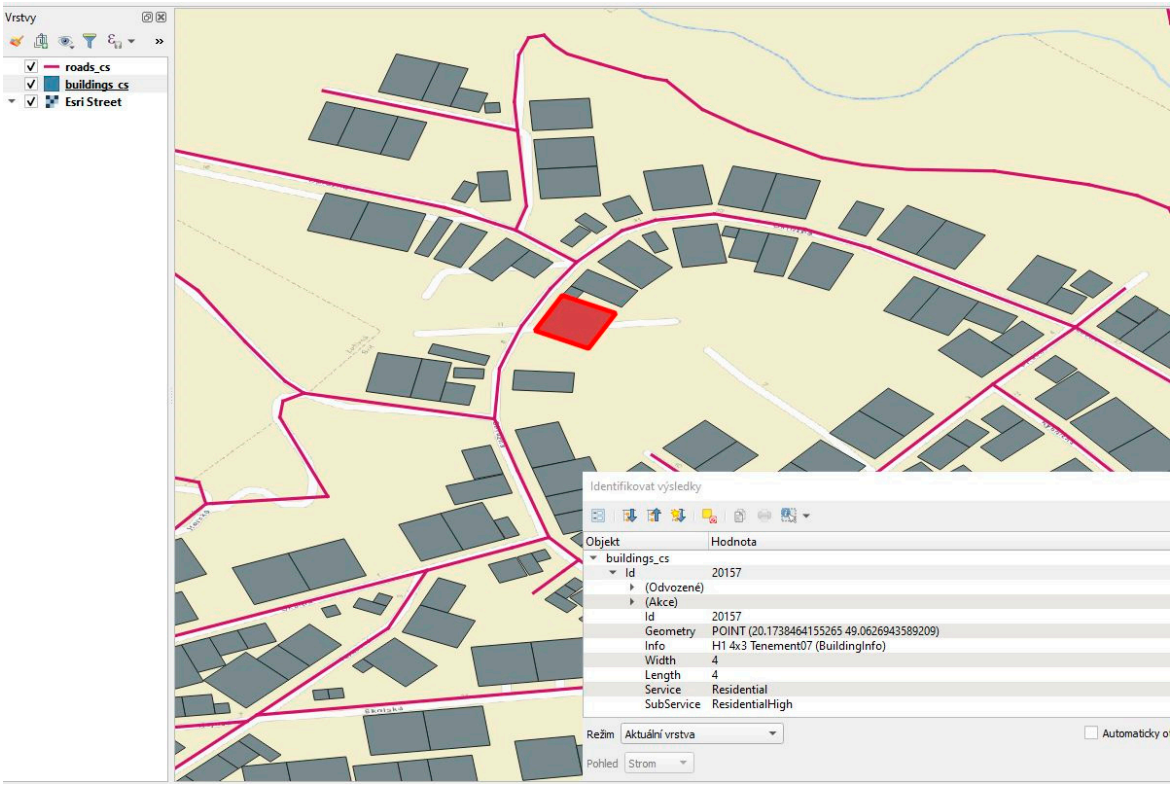

Figure 3. Game roads (purple) and game buildings (grey) exported as comma separated values (CSV) files and displayed in QGIS. Characteristics of the selected game building are displayed in the attribute viewer. The base map layer was added for geospatial context.

3. Results

We tested the presented methods by building models of the cities below in the game Cities: Skylines:

- Svit, Slovakia; and
- Olomouc, Czech Republic.

3.1. City of Svit

The city of Svit (Slovakia), with a current population of 7790 inhabitants, was established in 1934 and thus is relatively young. This city was built as a "company town", according to urban planning practices of that time. We recognized that these modern urban planning practices were similar to the simulations of the game; therefore, we decided to select this town for study. Due to its reasonable size, this town was also primarily used for testing the import and export methods. Coordinates of the bounding box represent an area of 17.28 × 17.28 km for the selected location and are shown in Table 3.

Table 3. Bounding box for the Svit area.

Point	Longitude	Latitude
Top right	20.311757266359	49.1351685440791
Bottom left	20.060024494391	48.9909768926872
Mid-point	20.1857094521613	49.063148018262

To create a base model of Svit in Cities: Skylines, the following methods were completed:

- Uploading the terrain using the game's inbuilt option. A heightmap for the area was obtained from Terrain.party. Instead of using a map box only restricted to a 17 km × 17 km box, we called the Terrain.party's API directly with exact coordinates:
    - http://terrain.party/api/export?name=Svit&box=20.311757266359,49.1351685440791,20.060024494391,48.9909768926872
- ImportRoads() to generate the road network in the game. OSM data with the tag "highway" were used to prepare the roads_rwo.csv file.
- ImportWaterWays() to generate water basins for waterways in the game. OSM data with the tag "waterway" were used to prepare the waterway_rwo.csv file.
- ImportWaterBody() to generate water basins for resources of standing water in the game. OSM data with the tag "natural=water" were used to prepare the water_rwo.csv file.
- ImportTreesRaster() to generate tree coverage in the game. The input raster image was prepared from the CORINE land cover layer. This layer was clipped according to the defined area and filtered to include only forested areas (codes 311, 312, 313). Forested areas covered a large part of the overall modeled area, resulting in breaching the limit of trees created (250,000). Therefore, the variable ImportTreesRasterMultiply was set to −2, meaning that every second creation of a tree will be skipped (i.e., dividing the total number of trees by 2).
- ImportServices() to generate services in the game. OSM data with the tag "amenity" were used to prepare the amenity_rwo.csv file.

Figure 4 displays the creation of Svit's base model in the game's map editor by calling the above GeoSkylines methods.

Figure 4. Creation of the base model of Svit from geographical data using the GeoSkylines methods. (**A**) An empty map after opening the map editor, (**B**) the map after uploading the terrain, (**C**) the map after generating the road network, and (**D**) the map after generating basins with water sources and tree coverage.

As part of the manual post-processing of the base model, the following methods were completed:

- Fixing issues of the created base model (e.g., due to incorrect geographical data);
- Adding water resources to created water basins;
- Adding outside highway connections to the created road network in order for the inhabitants to move into the city; and
- Adding zoning by using the game mod Image Overlay, which allows the display of pictures as a transparent layer in the game. A map image was created from Svit's zoning layer.

Figure 5 displays the resulting playable model of Svit after manual post-processing of the base model. Figure 6 displays a closeup of the playable model.

Figure 5. Overview of the resulting playable model of Svit after manual post-processing of the base model.

Figure 6. A closeup of the resulting playable model of Svit in Cities: Skylines.

3.2. City of Olomouc

The city of Olomouc (Czech Republic) is the sixth biggest city in the country, with a population of 100,378 inhabitants. This city is very old, especially the center, which contains many churches and other historic buildings. However, the city of Olomouc has gone through significant changes in the last few decades [38]. This city was selected for this study for its history and size. Coordinates of the bounding box representing an area of 17.28 × 17.28 km for the selected location are shown in Table 4.

Table 4. Bounding box for the Olomouc area.

Point	Longitude	Latitude
Top right	17.4136369459021	49.6855858352729
Bottom left	17.1674205694059	49.535040086535
Mid-point	17.2903398559444	49.6103605031655

To create a base model of Olomouc in Cities: Skylines, the following methods were completed:

- Uploading terrain using the game's inbuilt option. A heightmap for the area was prepared in GIS software from the Czech Republic's national DEM DMR5G.
- ImportRoads() to generate the road network in the game. OSM data with the tag "highway" were used to prepare the roads_rwo.csv file.
- ImportWaterWays() to generate water basins for waterways in the game. OSM data with the tag "waterway" were used to prepare the waterway_rwo.csv file.
- ImportWaterBody() to generate water basins for resources of standing water in the game. OSM data with the tag "natural=water" were used to prepare the water_rwo.csv file.
- ImportTreesRaster() to generate tree coverage in the game. An input raster image was prepared from the Urban Atlas Street Tree layer. This layer was clipped according to the defined area.
- ImportServices() to generate services in the game. OSM data with the tag "amenity" were used to prepare the amenity_rwo.csv file.

Figure 7 displays the base model of Olomouc generated in the game's map editor by calling the above GeoSkylines methods.

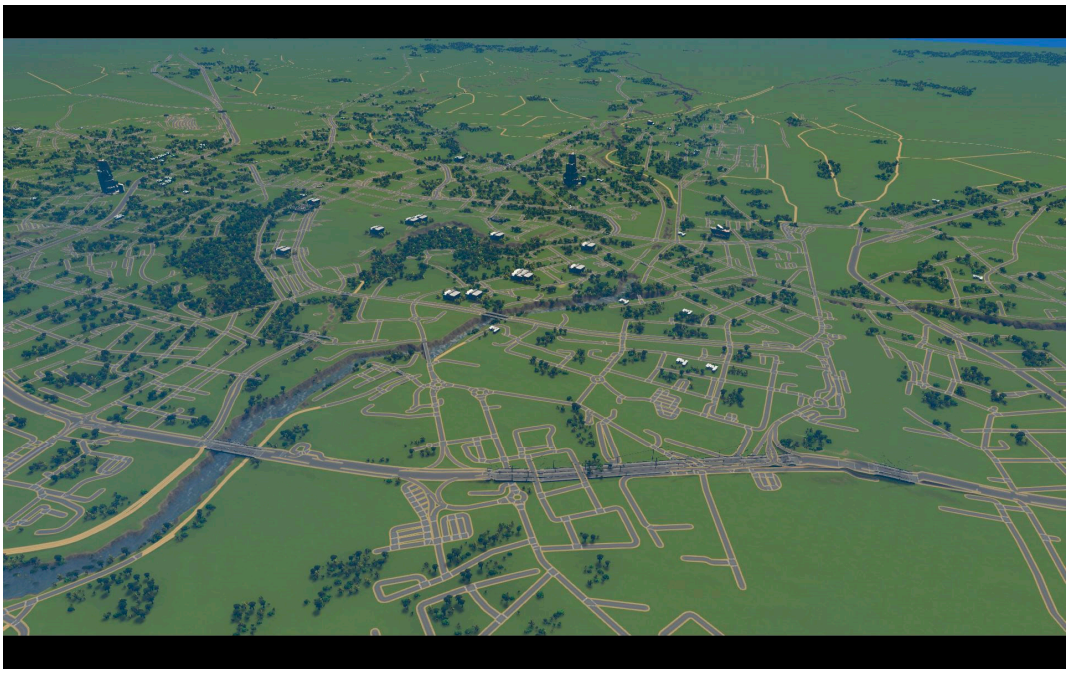

Figure 7. Base model of Olomouc in Cities: Skylines generated by calling GeoSkylines methods.

As part of the manual post-processing of the base model, the following methods were completed:

- Fixing issues of the created base model (e.g., due to incorrect geographical data);
- Adding water resources to the created water basins;
- Adding outside highway connections to the created road network in order for the inhabitants to move into the city;
- Adding zoning by using the game mod Image Overlay. A map image was created from Olomouc zoning layer; and
- Adding unique buildings such as churches and skyscrapers.

Figures 8 and 9 show the resulting playable model of the city center of Olomouc after manual post-processing of the base model.

Figure 8. A playable model of Olomouc in Cities: Skylines (historic center).

Figure 9. A playable model of Olomouc in Cities: Skylines (river view).

Aside from the physical visualizations, we also explored several game simulations in the created model of Olomouc. Figure 10 displays the simulated noise pollution in the Olomouc model. According to the game's algorithms, noise pollution is affected by traffic, industry, commercial zones, and specific types of buildings such as power plants.

Figure 10. Simulated noise pollution in the Cities: Skylines' model of Olomouc.

Inhabitants of the model of Olomouc utilize the road system to travel to work, school, shops, etc. The game individually tracks the passage of every citizen's vehicle as well as service and freight vehicles. In the traffic info view, the player can observe the traffic flow of each road and identify problematic parts of the road system. Figure 11 displays the traffic info view for the created model of Olomouc.

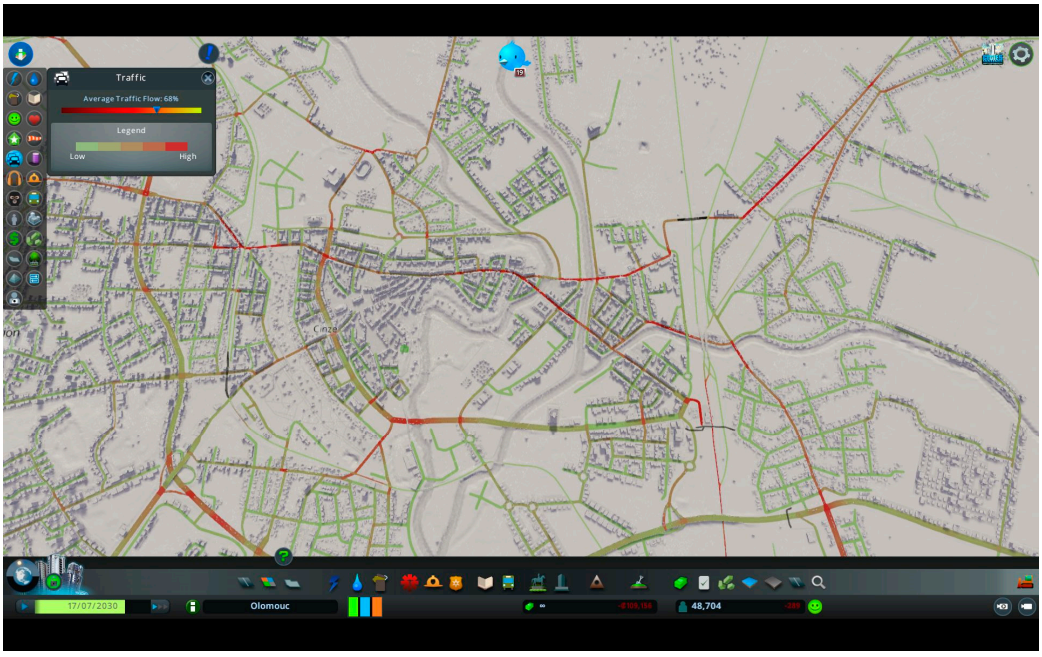

Figure 11. Traffic info view of the Cities: Skylines' model of Olomouc.

Although the average traffic flow is at an acceptable rate, there are several problematic parts where the traffic congestion is high. Examples of high congestion areas are Masarykova třída, Komenského, and Chválkovická Streets, or the intersection at Náměstí Hrdinů. For comparison, Figure 12 displays Environmental Systems Research Institute's (ESRI) World Traffic Service layer (source data provided by HERE Maps) where we can observe the authentic traffic situation in Olomouc. This layer identifies the same problematic parts of the Olomouc road network as the ones identified by the game's simulations.

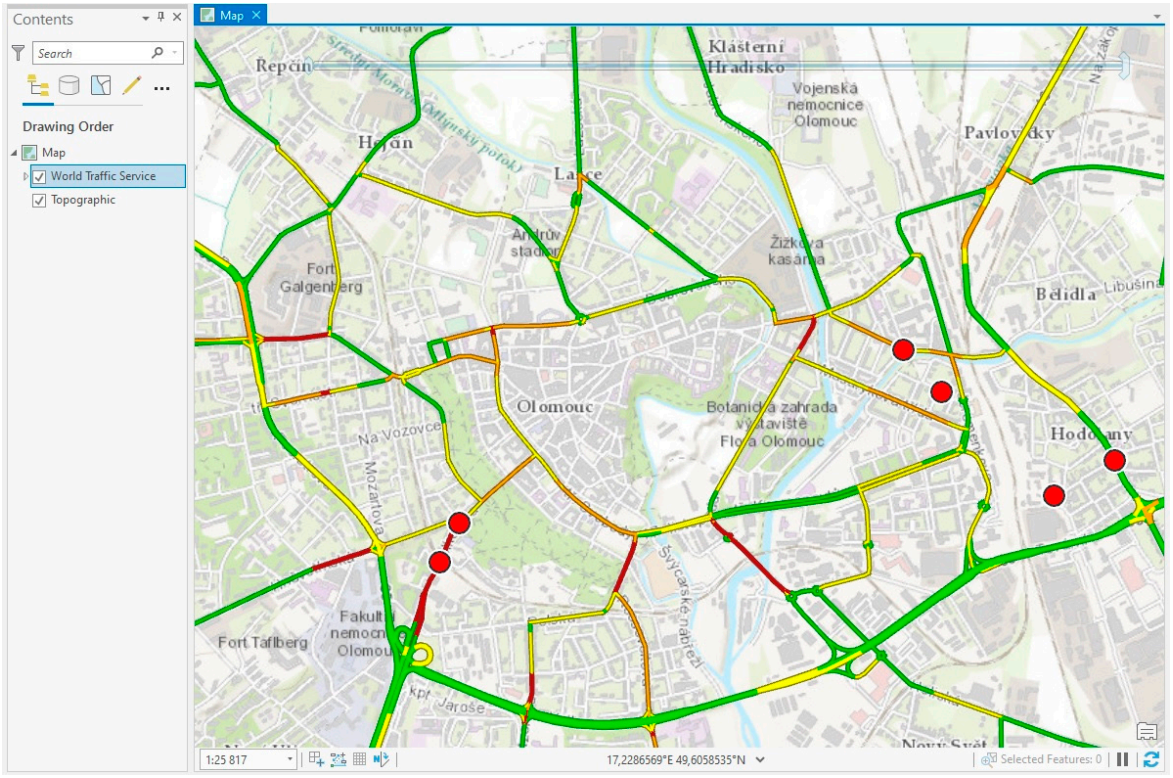

Figure 12. Environmental Systems Research Institute's (ESRI) World Traffic Service layer displaying the traffic situation in Olomouc.

The GeoSkylines game mod's code, detailed documentation for running the import and export methods, examples of configuration and input CSV files as well as all of the resulting base and playable models are available on development platform GitHub [37].

4. Discussion

Cities: Skylines is currently the most advanced city-building game on the market. This game offers features such as a visually appealing 3D graphics environment, a mass transit system, and agent-based simulations that present city related processes somewhat realistically. Due to these possibilities, Cities: Skylines has begun to be used for non-entertainment purposes. In all of the case studies identified, a model of a real-world place was created from geographical data. The resulting model was then used either for participation, education, or urban modeling projects [9–12]. However, the model creation process has been described as complicated and lengthy [12]. The lack of tools processing geodata for the purposes of Cities: Skylines is preventing a wider use of the game in non-entertainment ways.

Therefore, utilizing the game's modding API, we developed simple to use import methods to create a base model in Cities: Skylines accurate, fast, automated, and repeatable by using prepared geodata. During the development of the presented methods, maximum focus was given to enable the creation of as many game objects as possible with the highest achievable accuracy. The presented methods created road and rail networks, water basins, a tree coverage, zones, and services. An accurate conversion of the geographical coordinates of the source data to the game coordinates of the game

objects was achieved by implementing a standard conversion algorithm between the coordinate systems WGS 84 and UTM and then by simple recalculation of the UTM coordinates into the game coordinates (and vice versa). The conversion accuracy is confirmed in Figure 3, where the game objects, exported from the model of Svit and displayed in GIS software, properly aligned with the added base layer.

However, to create a playable model, manual post-processing of the base model is required. This process might include:

- Fixing issues of the base model (e.g., due to incorrect source geographical data or the code's inability to efficiently generate complex parts of the model such as tunnels).
- Adding necessities such as connecting the city to a highway for new inhabitants to move in or adding water resources.
- Zoning in the case where GeoSkylines's method ImportZones() cannot be used (e.g., vector layer of zoning for the selected location is not available).
- Adding unique buildings such as churches and skyscrapers.
- Adjusting the game simulations by implementing selected game mods.

The amount of time spent on manual post-processing of the base model depends on the size of the model and required level of detail. Adding the necessities so that the model is playable can take tens of minutes. However, for a large model where a great level of detail is required, the manual post-processing can take hours to complete. To mitigate the amount of time spent on the manual post-processing, it is recommended that this activity is completed by an experienced Cities: Skylines player. The fact that manual post-processing is still required for the creation of the playable model in Cities: Skylines is a limitation. Nevertheless, by employing the presented methods, the complexity and the time spent on the creation of such a model decreases significantly.

Using the presented methods, we built playable models of the cities of Svit (Slovakia) and Olomouc (Czech Republic). The creation of the playable model of Svit took approximately 40 min because the modeled city is relatively small, and the planned level of detail of the model was set to low. The creation of the playable model of Olomouc took approximately 8 h because the city is much larger than Svit, and the planned level of detail of the model was set higher than in the case of Svit.

When creating roads—either by script or manually—the game automatically creates zone blocks along them. These zone blocks can then be assigned with one of the available zones in the game: residential low, residential high, commercial low, commercial high, office and industrial. After setting the zones, the game starts to automatically construct the buildings. This automated creation of buildings speeds up the overall model creation, on the other hand, the look of the resulting model is more generalized. The game offers many features to increase the visual accuracy of the model. Among others, unique buildings such as churches or skyscrapers can be added to the model. Unique buildings can be created in the game's asset editor or imported from 3D modeling software. Selected game mods allow the player to construct each individual building. Although all these options are very time consuming, we recommend utilizing them if the purpose of the model in Cities: Skylines is the visualization of a city or part of it.

If, however, the purpose of the model is to run simulations, then a high level of detail of visual accuracy is not necessary as the looks of the buildings and other game objects do not affect the simulations. In some cases, the visual accuracy can even be contradictory to the game simulations. For example, the university complex in Olomouc consists of several buildings, but the university in the game is represented by a single building. Hence, adding another university building to the game to comply with the visual accuracy would affect the simulation logic. Uploaded terrain also has a minimal effect on the game simulations and serves rather as an aesthetic feature. The created model of Olomouc consisted of parts where the visual accuracy was relatively high as well as parts where the visual accuracy was reduced in favor of the game simulations.

While playing the more detailed model of Olomouc, we encountered several limitations and unrealistic behavior of the game. The demand for some of the services such as fire stations was exaggerated in the game. There are two fire stations in Olomouc, and in reality, this is sufficient. However, the same number of fire stations in the created model of Olomouc were not enough to satisfy the exaggerated demand in the game, resulting in frequent fires. Fortunately, this and similar behavior can be mitigated by game mods (e.g., the game mod No Fires, which stops the fire simulation altogether). On the other hand, traffic simulations provided satisfying results. The game's traffic simulation identified the same problematic parts of the Olomouc' road network as the ESRI's World Traffic Service layer, which presents authentic traffic data. The possibilities of fine-tuning the model simulations are vast and should be addressed in future research

The presented GeoSkylines game mod also provides methods for exporting game objects, specifically: road and rail networks, buildings, zones, and trees. Thanks to this feature, player creations in Cities: Skylines can be output as GIS data, which can then be processed further in professional software. By employing the export methods, the game could be used as a data collection tool in a participation project similar to the case study of the city of Hämeenlinna where players were asked to design a new suburb [9].

Other uses of the presented methods and the overall model creation process might involve implementing them into urban planning classes. The use of city-building games for education purposes is predominantly dedicated to SimCity; the latest implementation of SimCity was done by Terzano and Morckel [16]. However, Cities: Skylines has also begun to be experimented with in education [30]. Using the presented methods, a basic model of the selected city can be created, and this model then presented to students. By playing this model, students can apply their ideas of urban planning, as in a study done by Kim and Shin [39], where students created imaginary cities in SimCity.

5. Conclusions

The city-building game Cities: Skylines has begun to be used with for visualizations of real-world places in various participation or education projects. However, there is a current lack of tools that process geographical data for the purposes of the game, thus the model creation has been described as complicated and time consuming. The main aim of the presented research was to develop a tool that will help bridge the gap between the game Cities: Skylines and the GIS domain.

Programmed in the game's API, the game mod GeoSkylines offers import methods that create road and rail networks, water basins, a tree coverage, zones, and services. An accurate conversion of the geographical coordinates of the source data to the game coordinates of the game objects was achieved by implementing a standard conversion algorithm between the coordinate systems WGS 84 and UTM and then by simple recalculation of the UTM coordinates into the game coordinates (and vice versa). The game mod GeoSkylines also provides methods to export game creations into GIS data that can be further processed using professional software.

Using the presented methods, geographically accurate base models of the cities of Svit (Slovakia) and Olomouc (Czech Republic) were created in the game Cities: Skylines. However, even with the use of the presented methods, manual post-processing of the base models was required in order to make the models playable. To mitigate this limitation, the manual post-processing was completed by an experienced Cities: Skylines player.

Additionally, selected simulations of the more detailed model of Olomouc were explored. In several cases, the game simulations were unrealistic (e.g., frequent fires due to the game's exaggerated demand for fire services). To mitigate unrealistic behavior of some simulations, selected game mods were implemented. On the other hand, some simulations such as the traffic simulations of citizens', service and freight vehicles provided satisfying results. The possibilities of the Cities: Skylines simulations on models of a real-world place are vast and are planned to be explored in future research.

All of the developed methods have been bundled into our Cities: Skylines game mod named GeoSkylines. The code of this mod along with the example CSV files and models are freely available on development platform GitHub [37].

Author Contributions: Conceptualization, Jan Piňos; Methodology, Jan Piňos and Vít Voženílek; Data collection Jan Piňos and Ondřej Pavliš; Investigation Jan Piňos and Ondřej Pavliš; Writing—original draft preparation, Jan Piňos and Ondřej Pavliš; Writing—review and editing, Jan Piňos and Vít Voženílek; All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by the Grant Agency of Palacky University Olomouc (Grant No.: IGA_PrF_2019_014).

Acknowledgments: This paper was created within the project "Innovation and application of geoinformatic methods for solving spatial challenges in the real world." (IGA_PrF_2019_014) with the support of the Internal Grant Agency of Palacky University Olomouc.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Anderson, M.; Jiang, J. Teens, social media & technology 2018. *Pew Res. Cent.* **2018**, *31*, 2018.
2. Gee, J.P. What Video Games have to Teach Us about Learning and Literacy. *Comput. Entertain.* **2003**, *1*, 20. [CrossRef]
3. Prensky, M. Digital game-based learning. *Comput. Entertain.* **2003**, *1*, 21. [CrossRef]
4. Gordon, E.; Schirra, S.; Hollander, J. Immersive planning: A conceptual model for designing public participation with new technologies. *Environ. Plan. B Plan. Des.* **2011**, *38*, 505–519. [CrossRef]
5. Poplin, A. Playful public participation in urban planning: A case study for online serious games. *Comput. Environ. Urban Syst.* **2012**, *36*, 195–206. [CrossRef]
6. Using MINECRAFT for Community Participation. Available online: https://unhabitat.org/books/manual-using-minecraft-for-community-participation/ (accessed on 21 April 2019).
7. Moss, R. From SimCity to, well, SimCity: The history of city-building games. *Ars Tech.* **2015**, *11*, 1–3.
8. Roy, J. From Video Game to Day Job: How 'SimCity' Inspired a Generation of City Planners. *LA Times*. 2019. Available online: https://www.latimes.com/business/technology/la-fi-tn-simcity-inspired-urban-planners-20190305-story.html (accessed on 5 March 2019).
9. Guzman, J. Finland City Holds City Planning Contest Using Video Game. Available online: https://www.rappler.com/technology/news/123474-finland-hameenlinna-cities-skylines-planning-contest (accessed on 12 March 2019).
10. Video Game Cities Skylines Helps Plan Stockholm Development. Available online: https://www.bbc.com/news/av/39200838/video-game-cities-skylines-helps-plan-stockholm-development (accessed on 8 March 2019).
11. Grande, T.R. Se Studentene Bygge Fremtidens Oslo! Available online: https://www.nmbu.no/aktuelt/node/34528 (accessed on 10 March 2019).
12. Juraschek, M.; Herrmann, C.; Thiede, S. Utilizing gaming technology for simulation of urban production. *Procedia CIRP* **2017**, *61*, 469–474. [CrossRef]
13. Rufat, S.; Ter Minassian, H. Video games and urban simulation: New tools or new tricks? *Cybergeo Eur. J. Geogr.* **2012**. [CrossRef]
14. Devisch, O. Should planners start playing computer games? Arguments from SimCity and Second Life. *Plan. Theory Prac.* **2008**, *9*, 209–226. [CrossRef]
15. D'Artista, B.R.; Hellweger, F.L. Urban hydrology in a computer game? *Environ. Model. Softw.* **2007**, *22*, 1679–1684. [CrossRef]
16. Jabareen, Y.R. Sustainable urban forms: Their typologies, models, and concepts. *J. Plan. Educ. Res.* **2006**, *26*, 38–52. [CrossRef]
17. Starr, P. Seductions of Sim: Policy as a simulation game. *Am. Prospect* **1994**, *17*, 19–29.
18. Bereitschaft, B. Gods of the city? Reflecting on city building games as an early introduction to urban systems. *J. Geogr.* **2016**, *115*, 51–60. [CrossRef]
19. Haahtela, P. *Gamification of Education: Cities Skylines as an Educational Tool for Real Estate and Land Use Planning Studies*; Aalto University Learning Centre: Espoo, Finland, 2015.

20. Batty, M.; Torrens, P.M. Modelling complexity: The limits to prediction. In Proceedings of the 12th European Colloquium on Quantitative and Theoretical Geography, Saint-Valery-en-Caux, France, 7–11 September 2001.
21. Fulton, K. Cities: Skylines CEO: We're Not Changing Traffic (But Natural Disasters Would Be Cool). Available online: https://www.techradar.com/news/gaming/cities-skylines-ceo-zombies-are-cool-but-leave-natural-disasters-to-us-1297846 (accessed on 10 March 2019).
22. Scacchi, W. Computer game mods, modders, modding, and the mod scene. *First Monday* **2010**, *15*, 5. [CrossRef]
23. Sotamaa, O. When the game is not enough: Motivations and practices among computer game modding culture. *Games Cult.* **2010**, *5*, 239–255. [CrossRef]
24. Cities: Skylines Celebrates Fourth Anniversary and Six Million Copies Sold. Available online: https://bit.ly/2EV2CtJ (accessed on 9 March 2019).
25. Devisch, O.; Arentze, T.; Borgers, A.W.J.; Timmermans, H.J.P. An agent-based model of residential choice dynamics in non-stationary housing markets. In Proceedings of the Paper CUPUM Conference, London, UK, 29 June–1 July 2005.
26. Portugali, J. *Self-Organization and the City*; Springer Science & Business Media: Berlin, Germany, 2012.
27. Adams, P.C. Teaching and learning with SimCity 2000. *J. Geogr.* **1998**, *97*, 47–55. [CrossRef]
28. Gaber, J. Simulating planning: SimCity as a pedagogical tool. *J. Plan. Educ. Res.* **2007**, *27*, 113–121. [CrossRef]
29. Minnery, J.; Searle, G. Toying with the city? Using the computer game SimCity™ 4 in planning education. *Plan. Prac. Res.* **2014**, *29*, 41–55. [CrossRef]
30. Terzano, K.; Morckel, V. SimCity in the community planning classroom: Effects on student knowledge, interests, and perceptions of the discipline of planning. *J. Plan. Educ. Res.* **2017**, *37*, 95–105. [CrossRef]
31. Matulef, J. SimCity Sold Over 2 Milion Copies. Available online: https://www.eurogamer.net/articles/2013-07-24-simcity-sold-over-2-million-copies (accessed on 25 July 2013).
32. r/CitiesSkylines. Available online: https://www.reddit.com/r/CitiesSkylines/ (accessed on 6 July 2019).
33. r/SimCity. Available online: https://www.reddit.com/r/SimCity/ (accessed on 6 July 2019).
34. Info Views. Available online: https://skylines.paradoxwikis.com/Info_views (accessed on 10 June 2019).
35. Eisele, S.; Mardari, I.; Dubey, A.; Karsai, G. Riaps: Resilient information architecture platform for decentralized smart systems. In Proceedings of the 2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC), Toronto, AB, Canada, 16–18 May 2017; pp. 125–132.
36. Amazingly Detailed Metropolises Recreated in Cities: Skylines—In Pictures. Available online: https://www.theguardian.com/cities/gallery/2015/jul/15/bulldoze-white-house-real-cities-cities-skylines-in-pictures (accessed on 12 March 2019).
37. GeoSkylines game mod. Available online: https://github.com/gonzikcz/GeoSkylines/ (accessed on 30 December 2019).
38. Burian, J.; Brus, J.; Voženílek, V. Development of Olomouc city in 1930–2009: Based on analysis of functional areas. *J. Maps* **2013**, *9*, 64–67. [CrossRef]
39. Kim, M.; Shin, J. The pedagogical benefits of SimCity in urban geography education. *J. Geogr.* **2016**, *115*, 39–50. [CrossRef]